PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2000-311129
(43)Date of publication of application : 07.11.2000

(51)Int.Cl. G06F 13/00
G06F 3/12
G06F 15/16
G06F 15/177

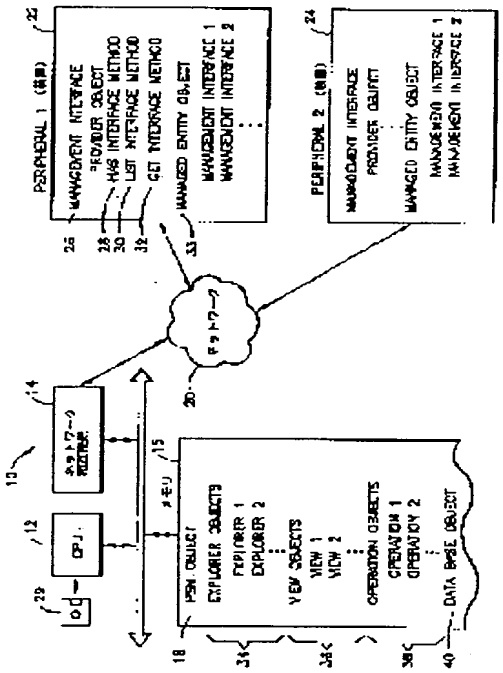
(21)Application number : 2000-079686
(22)Date of filing : 22.03.2000

(71)Applicant : HEWLETT PACKARD CO <HP>
(72)Inventor : SIMPSON SHELL S
CLOUGH JAMES
GARTNER M SCOTT

(30)Priority
Priority number : 99 298590 Priority date : 23.04.1999 Priority country : US

(54) PERIPHERAL DEVICE MANAGEMENT SYSTEM

(57)Abstract:
PROBLEM TO BE SOLVED: To interact with devices of different types of a peripheral system by actuating an interface provider object so as to restore a management interface permitting access to information regarding an object device to be managed.
SOLUTION: A client computer 10 is equipped with a CPU 12, a network interconnection module 14, and a memory 16 which has a PSM object 18. Peripheral devices 22 and 24 to be managed by the client computer 10 are provided with management interface provider objects 26, each of which are enabled to access one or more MIs(management interface) which makes it possible to retrieve information regarding the roles of the respective peripheral devices 22 and 24.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

FP05127-IDS

(11) Publication number : Japanese Patent Laid-Open No. 2000-311129
(43) Date of publication of application : November 7, 2000

[0008]
Network software must interact with physical devices that are connected to the network. In the past this has been accomplished by describing a physical topology and assigning each machine a name. However, since physical topologies vary from site to site, software deployed at many locations cannot be described in this way. This invention, rather than specifying particular machines, uses a logical topology that describes the roles (or services) that machines play/provide. Examples of roles that machines play in peripheral system management (PSM) software embodying the invention are given below:
*Management Client Machine: the role of a machine that presents a user interface for PSM software. For example, the role of management client is performed by whatever machine runs a web browser that is used to access a network administration program.
*Management Server Machine: the role of a machine that provides logic for PSM software. For example, the server machine on which a network administration program is installed performs the role of a management server machine.
*Peripheral Client Machine: the role of a machine making use of services provided by a peripheral. An end-user workstation making use of a printer is an example of a peripheral client machine.
*Peripheral Server Machine: the role of a machine mediating access to the services provided by a peripheral device. A Windows NT (a trademark of the Microsoft Corporation) machine that shares its printers is an example of a peripheral server machine.
*Peripheral Machine: the role of a machine that serves as a peripheral. Printers and scanners are examples of peripheral devices.
*Peripheral Directory Machine: the role of a machine that provides a service used to find peripheral machines and peripheral server

machines.

[0019]
FIG. 1 illustrates a typical ME object with references to MIs that enable retrieval of feature information of the device that provides the role represented by the ME. Each MI contains one or more functions, some of which may enable retrieval of information from the device and others of which may enable the associated device to perform a function. Again, it is to be recalled that the ME corresponds to a role performed by a device, and not necessarily to the device as a whole--unless the device only performs one role.

[0038]
InstallFont Operation
Consider the following interface used to manage true type fonts:

[0039]
[Description 1]
interface TrueTypeFontManagement:

Management Interface

{ void Add(Font f); void FontList List( ); void Remove(FontListElement f); };

[0040]
Now, consider a peripheral systems management application that installs fonts on end-user machines (e.g., PCs) and printers. Such an application can be constructed using an Operation that installs a font on a machine supporting the TrueTypeFontManagement interface. The application simply identifies all the MEs supporting the TrueTypeFontManagement interface, configures the InstallFont Operation with the desired font file, and invokes the Operation on the MEs identified earlier.

[0041]

AssignSequentialNetworkAddresses Operation

When changing a physical network from one subnet to another, it is necessary to renumber all the devices on the physical network. Network administration programs support mechanisms for reassigning network addresses sequentially to all the devices (managed by those applications) on the physical network. This can be accomplished using an Operation. Such an Operation is configured with a range of network addresses. The peripherals to renumber are then selected by the application (with or without user intervention depending on the application). Finally, the AssignSequentialNetworkAddresses Operation is invoked on the previously selected collection of peripheral MEs.

[0042]

Views

Definition

A View is an object that displays a collection of MEs using MIs. Each ME in the collection being displayed must support the MIs required by the View. Views have state and can be configured. A View is much like an Operation, except instead of operating on the supplied MEs, the MEs are displayed. A View can be broken into three parts: state, configuration, and display. The state keeps track of the current configuration and does not have a user interface. The configuration is similar to a configuration applet for an Operation. It is responsible for presenting a user interface that allows the state to be modified. The display is responsible for displaying a collection of MEs. The order of MEs may or may not be important depending on the View. Some Views may ignore the order of MEs in a collection and simply display the top three "interesting" MEs (and summarize the rest). Others might display all MEs in the order provided in the collection. More than one instance of a View can exist at the same time, with a different state.

[0043]

EXAMPLES

Consumables View

It is costly when a printer runs out of toner. Because users are often

untrained, changing a toner cartridge takes longer than if a technician replaced the cartridge. Untrained users can also accidentally damage the printer. To address this problem, it is useful to have a means of finding out which printers are almost out of toner, so the toner can be replaced before it is exhausted. This is accomplished using a View. Such a View displays printers below a certain toner threshold, ordered from least to most amount of toner. By changing the low toner threshold, the user can vary the number of printers displayed. All printers are displayed if the low toner threshold is set to 100%.

[0044]
Scanner View
This View displays MEs of a specific type. A View can be created for printers and would show the maximum number of pages per minute that the printer could deliver. A View for scanners can have a field indicating whether the scanner supports color. Accordingly, Views can be created to support specific types of MEs and display fields that make sense for that type.

(19)日本国特許庁（ＪＰ）　(12) 公開特許公報（A）　(11)特許出願公開番号
特開2000－311129
(P2000－311129A)
(43)公開日　平成12年11月７日(2000.11.7)

(51)Int.Cl.⁷		識別記号	FI		テーマコード（参考）
G06F	13/00	357	C06F	13/00	357A
	3/12			3/12	D
	15/16	620		15/16	620S
	15/177	670		15/177	670F

審査請求　未請求　請求項の数1　ＯＬ　（全 16 頁）

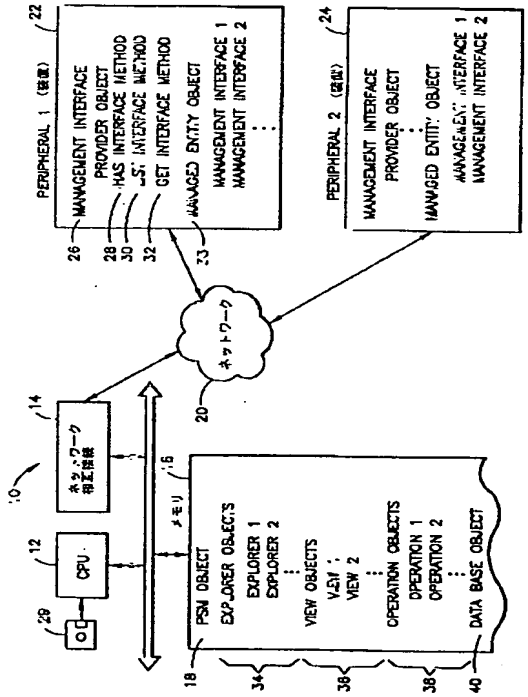
(21)出願番号　特願2000－79686(P2000－79686)
(22)出願日　平成12年３月22日(2000.3.22)
(31)優先権主張番号　０９／２９８５９０
(32)優先日　平成11年４月23日(1999.4.23)
(33)優先権主張国　米国（ＵＳ）

(71)出願人　398038580
ヒューレット・パッカード・カンパニー
HEWLETT－PACKARD COMPANY
アメリカ合衆国カリフォルニア州パロアルト　ハノーバー・ストリート　3000
(72)発明者　シェル・エス・シンプソン
アメリカ合衆国83713アイダホ州ボイジー、ノース・メイドストーン・ウェイ　5196
(74)代理人　100081721
弁理士　岡田　次生

最終頁に続く

(54)【発明の名称】　周辺装置管理システム

(57)【要約】
【課題】管理コンピュータが周辺システムの装置を管理することを可能にするオブジェクト指向システムを提供する。
【解決手段】管理対象周辺装置は、それが提供するサービスの各々に関連する"管理対象エンティティ(ME)"を持つ。各MEは、MEの機能に関する情報を提供し、MEを含む装置が１つの機能を実行することを可能にする複数のメソッドから構成される"管理インタフェース(MI)"オブジェクトに対する参照を含む。管理側コンピュータは、ユーザ照会に応答して、対象周辺装置に対応するMEを決定するためエクスプローラ・オブジェクトを起動し、その周辺装置、その活動およびユーザの照会に関連する情報および諸メソッドを含むMEに関連するMIオブジェクトを復元するためインタフェース・プロバイダ・オブジェクトのメソッドを起動することによって、所望の周辺装置の管理を行う。



【特許請求の範囲】
【請求項１】管理コンピュータが周辺システムの装置を管理することを可能にするシステムであって、
管理対象装置の各々が、必ずしも上記管理コンピュータではないコンピュータのため１つまたは複数のサービスを提供すると共に、インタフェース・オブジェクトに関するメソッドが実行されることを可能にする管理インタフェース・プロバイダ・オブジェクトを記憶し、
上記サービスの各々が、１つまたは複数の管理インタフェースに対する参照を含む関連する管理対象エンティティ・インタフェースを有し、
上記管理インタフェースの各々が、上記管理対象エンティティによって表され、上記管理対象装置によって実行される上記サービスの機能に関する情報を提供するかまたは上記管理対象装置が関数を実行することを可能にするか少なくともいずれかを実行する１つまたは複数のメソッドを備え、
上記管理コンピュータが、
上記管理対象装置の各々との通信を可能にする入出力手段と、
エクスプローラ・オブジェクトを記憶するメモリと、
管理対象装置のサービスに関するユーザ照会に応答して、上記管理対象装置のサービスに対応する管理対象エンティティを決定するため上記エクスプローラ・オブジェクトを起動し、また、上記照会に関連し上記管理対象装置に関する情報へのアクセスを可能にするかまたは上記管理対象装置が上記照会に関連する関数を実行することを可能にする上記管理対象エンティティに関連する管理インタフェースを復元するため上記インタフェース・プロバイダ・オブジェクトを起動するプロセッサ手段と、
を備える、周辺装置管理システム。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、周辺システムの管理に関するもので、特に、オブジェクト・インタフェースを利用する周辺システム管理に関するものである。
【０００２】
【従来の技術】周辺システムの管理は、周辺システムを構成する装置の構成、それら装置の状態の取得およびそれら装置からのイベントの受領を伴う。それら装置の能力は広範囲にわたり、また、それら装置と対話するため使用される通信プロトコルもまた広範囲にわたる。周辺システムを管理するソフトウェアは、この多様性に対応しなければならず、その結果しばしば非常に複雑である。周辺システムという用語は、本明細書において使用される場合、プリンタ、スキャナ、サーバ、多機能装置およびパーソナル・コンピュータのような周辺クライアントを含む。
【０００３】周辺装置、周辺サーバおよび周辺クライアントの新しいモデルが絶えず業者から出荷されるので、それは、周辺システム管理ソフトウェア・システムの開発者に一層の挑戦を与える。そのようなソフトウェアは、そのような新しい装置と新しい装置機能をサポートするため絶えず修正されなければならず、そのため、ソフトウェアの複雑性が一層増大する。周辺装置、周辺サーバおよび周辺クライアントによって提供される能力はそれらのそれぞれの構成によって変わる。ハードディスクのような資源が追加または削除されると、周辺システム管理は、そのような能力変更を取り扱わなければならず、そのため周辺システム管理ソフトウェアの複雑性が一層増大する。
【０００４】以下の記述において、周辺システム管理はその英語表現の"peripheral systemmanagement"を略称してＰＳＭと呼称される場合がある。ＰＳＭアプリケーションは、エンドユーザが各装置に同一のコマンドを繰り返して発信するのではなく単一のコマンドを使用して装置の集合に対してオペレーションを実行することを可能にする。例えば、ＰＳＭアプリケーションは、単一のコマンドの使用を通して装置の集合にアドレスを再割り当てするが、そのような装置の新しい能力には対処しない。また、ＰＳＭアプリケーションは、典型的には、管理対象装置の集合を表すユーザ・インタフェース・エレメントを表示する。このインタフェースを通して、ユーザは同時にいくつかの装置に対する管理活動を実行することができる。
【０００５】
【発明が解決しようとする課題】一般的に、このインタフェースは、拡張性がなく、ＰＳＭアプリケーションが複数装置を表示する方法を大幅に変更する能力を備えていない。対照的に、ＰＳＭアプリケーションは、装置設計者が装置をサポートする方法をカスタマイズする"アプレット"(すなわち小さいプログラム)の拡張を行うことを可能にする。ユーザが個々の装置を管理する要求を示す時、ユーザに装置固有のユーザ・インタフェースを提示するためアプレットが使用される。装置設計者が個々の装置を見る方法を拡張することができる場合と同様の方法で、装置の集合が提示される方法の変更を可能にする必要性が存在する。このように、必要とされる周辺システム管理システムは、周辺システムにおける複数のタイプの装置と対話することができ、同じ能力を備える装置グループに対して作用することができ、システムの１つのエレメントが特定の能力を備えているか否かを判断することができ、新たに組み込まれた能力に対してカスタマへの出荷の前および後いずれにおいても対処することができ、装置グループが提示される方法の変更に対処することができ、かつ、周辺装置システム・エレメントの能力を実行時に付与または削除することができるように十分に動的な周辺システム管理システムである。
【０００６】

【課題を解決するための手段】発明の課題を解決するため、管理コンピュータが周辺システムの装置を管理することを可能にするシステムが提供される。該システムにおいて、管理対象装置の各々は、必ずしも上記管理コンピュータではないコンピュータのため１つまたは複数のサービスを提供すると共に、インタフェース・オブジェクトに関するメソッドが実行されることを可能にする管理インタフェース・プロバイダ・オブジェクトを記憶し、上記サービスの各々は、１つまたは複数の管理インタフェースに対する参照を含む関連する管理対象エンティティ・インタフェースを有し、上記管理インタフェースの各々は、上記管理対象エンティティによって表され、上記管理対象装置によって実行される上記サービスの機能に関する情報を提供するかまたは上記管理対象装置が関数を実行することを可能にするか少なくともいずれかを実行する１つまたは複数のメソッドを備える。また、上記管理コンピュータは、上記管理対象装置の各々との通信を可能にする入出力手段、エクスプローラ・オブジェクトを記憶するメモリ、および、管理対象装置のサービスに関するユーザ照会に応答して、上記管理対象装置のサービスに対応する管理対象エンティティを決定するため上記エクスプローラ・オブジェクトを起動し、また、上記照会に関連し上記管理対象装置に関する情報へのアクセスを可能にするかまたは上記管理対象装置が上記照会に関連する関数を実行することを可能にする上記管理対象エンティティに関連する管理インタフェースを復元するため上記インタフェース・プロバイダ・オブジェクトを起動するプロセッサ手段を備える。

【０００７】本発明は、コンピュータが周辺システムの複数エレメントを管理することを可能にする。複数エレメントの各々は該コンピュータのため１つまたは複数のサービスを提供する。各周辺装置またはエレメントは、各サービスについて関連した"管理対象エンティティ"データ構造を持つ。"管理対象エンティティ"はその英語表記"managed entity"の頭文字を取って以下"ＭＥ"と略称される。各ＭＥデータ構造は、１つまたは複数の"管理インタフェース"オブジェクトに対する参照を含む。"管理インタフェース"はその英語表記"management interface"の頭文字を取って以下"ＭＩ"と略称される。ＭＩオブジェクトは、ＭＥの機能に関する情報を提供し、ＭＥを含む装置が１つの機能を実行することを可能にする１つまたは複数のメソッドから構成される。該コンピュータは、周辺システム装置の各々との通信を可能にする入出力モジュールおよびエクスプローラ・オブジェクトを記憶するメモリを含む。該コンピュータにおけるプロセッサは、周辺装置に関するユーザ照会に応答して、(i)その周辺装置に対応するＭＥを決定するためエクスプローラ・オブジェクトを起動し、(ii)その周辺装置、その活動およびユーザの照会に関連する情報および諸メソッドを含むＭＥに関連するＭＩオブジェクトを復元するためインタフェース・プロバイダ・オブジェクトのメソッドを起動する。

【０００８】

【発明の実施の形態】ネットワーク・ソフトウェアは、ネットワークに接続している物理装置と対話しなければならない。過去においては、これは、物理的トポロジを記述して各マシンに名前を割り当てることによって達成された。しかし、物理トポロジはサイトごとに変わるので、多数の位置に展開されるソフトウェアはそのように記述されることができない。本発明は、特定のマシンを指定するのではなく、マシンが演ずるまたは提供する役割またはサービスを記述する論理トポロジを使用する。本発明を実施する周辺システム管理(ＰＳＭ)ソフトウェアにおいてマシンが演ずる役割の例は次の通りである。

＊管理クライアント・マシン：ＰＳＭソフトウェアのためユーザ・インタフェースを提供するマシンの役割。例えば、管理クライアントの役割は、ネットワーク管理プログラムにアクセスするために使用されるウェブ・ブラウザを実行させるどのようなマシンによっても演じられる。

＊管理サーバ・マシン：ＰＳＭソフトウェアのため論理を提供するマシンの役割。例えば、ネットワーク管理プログラムが導入されているサーバ・マシンは管理サーバ・マシンの役割を実行する。

＊周辺クライアント・マシン：周辺機器によって提供されるサービスを利用するマシンの役割。プリンタを利用するエンドユーザ・ワークステーションは周辺クライアント・マシンの１例である。

＊周辺サーバ・マシン：周辺装置によって提供されるサービスに対するアクセスを調停するマシンの役割。プリンタを共有するウインドウズ（登録商標）ＮＴマシンは周辺サーバ・マシンの１例である。

＊周辺マシン：周辺機器の役をつとめるマシンの役割。プリンタおよびスキャナは周辺装置の例である。

＊周辺ディレクトリ・マシン：周辺マシンおよび周辺サーバ・マシンを見つけ留ために使用されるサービスを提供するマシンの役割。

【０００９】１つの物理マシンが複数の役割を実行し複数のサービスを提供することができる点注意する必要がある。例えば、中央サーバは、管理サーバ・マシン(ネットワーク管理プログラムが導入されているマシン)と周辺サーバ・マシン(例えばプリント・サーバ)の両方の役割を果たすことができる。本発明を取り入れる周辺管理ソフトウェアは、周辺機器に関連するＭＥを検出、構成、表示、修正および監視することができる。ＭＥは、装置の役割および装置が提供するサービスがそれを通して管理されることができるソフトウェア・オブジェクトである。上述のように、単一の装置は複数の役割を持つことができるので、複数ＭＥに関連づけられることができる。

【００１０】管理対象エンティティ(ME)の特徴
・MEは、管理対象をモデル化するために使用されるアーキテクチャ・アブストラクトである。
・MEは、エクスプローラのアーキテクチャ・アブストラクトによって発見される。
・MEは、データベースのアーキテクチャア・アブストラクトによって構成される。
・MEは、ビユーのアーキテクチャ・アブストラクトによって表示される。
・MEは、オペレーションのアーキテクチャ・アブストラクトによって修正される。
・MEは、イベント(Event)および行動(Action)のアーキテクチャ・アブストラクトを使用して監視される。
【００１１】以下の説明から理解されることであろうが、上記識別されたアブストラクトの各々は、古典的な意味において"変数"の状態を維持し、１つまたは複数のメソッドでその動作を実行するソフトウェア・オブジェクトとして構成される。以下に記述される種々のオブジェクトによって、本発明を取り入れるＰＳＭソフトウェアは、システムおよびソフトウェア修正の適応に対する大幅な柔軟性を与えられる。
【００１２】管理対象エンティティ(ME)
定義
MEは、周辺管理問題ドメインにおいてユーザが認識できるエレメントを表すオブジェクトである。MEは、１つまたは複数の管理インタフェース(ＭＩ)の集合を提供する。ＭＩは、MEと対話(またはMEを識別)するために使用される１つまたは複数の関数セットである。
【００１３】プリンタを管理する責任があるユーザは、プリンタ、プリント・サーバおよびプリント・クライアントの存在を認識する。ＰＳＭソフトウェアの有無にかかわらず、このユーザは、問題を解決するため、これらのエレメントの各々と対話する。同じこのユーザは、スプーラの詳細なアーキテクチャについて知識をたぶん持たないであろう。ポート・モニタまたはプリント・プロセッサの存在もおそらくそのようなユーザと無関係であろう。問題を解決する際、このユーザは、よく知っているエレメントだけに関心を持つ。そのようなよく知られていて容易に認識されるエレメントが、ＰＳＭソフトウェアにおいてユーザが見つけることを期待する管理の基本単位である。このような管理の基本単位を表すオブジェクトがMEと呼ばれる。
【００１４】これら管理基本単位の各々は、付加的エレメントに再分割される。例えば、プリント・サーバはスプーラから構成され、更にこのスプーラがドライバ、ポート・モニタおよびプリント・プロセッサのように一層多くのエレメントから構成される。これらのエレメントの各々が管理される必要があるにもかかわらず、それらは典型的にはプリント・サーバの管理という文脈において管理される。ユーザは、プリント・サーバと対話することによって特定のプリント・サーバ上のドライバをアップグレードする。
【００１５】MEによって提供されるＭＩの集合は拡張性がある。例えば、プリンタは、たたえ対象となるインタフェースがＰＳＭソフトウェアにすでに認識されていないとしても、(追加または削除のような)プリンタの機能を管理するためのインタフェースを提供することができる。インタフェースを使用するため、オペレーションおよびビユー・オブジェクトが提供される(後述)。また、MEによって提供されるＭＩの集合は動的である。ＭＩ集合は実行時に修正されることができる。MEの変更に対する資源が使用可能であれば、そのMEによって提供されるＭＩは変わることができる。例えば、プリンタがハードディスクを持つことも持たないこともある。ハードディスクの存在が大容量記憶フォントを管理することが可能か否かを左右するので、ハードディスクが使用可能でなければ、大容量記憶フォントの管理に関連するＭＩは提供されない。
【００１６】特定のMEによってサポートされるＭＩ(およびそれらＭＩが含む関数)は実行時に照会されることができる。ＭＩを照会するアプリケーションは、必ずしもそれらＭＩを使用する方法を知る必要はない。むしろ、アプリケーションは、各MEがどのＭＩを提供し、どのＭＩがこの情報を使用してこれらインタフェースを理解するコンポーネントとＭＩを一致させるか単に追跡することができる。例えば、フォントを管理するために使用されるＭＩについて何も知らないアプリケーションでも、フォント管理インタフェースを必要するオペレーションをそのインタフェースを供給するMEと一致させることができる。
【００１７】管理インタフェース
通常、１つのオブジェクトは、単一の静的インタフェースを持つ。しかしながら、１つのインタフェースだけを持つということは、オブジェクトの能力が変更される都度インタフェースが修正されなければならないことを意味する。同じインタフェースの複数のバージョンを持つことは混同をもたらす。本発明において、複数インタフェースを持つ１つのMEは、各々が管理インタフェース(ＭＩ)と呼ばれる１つまたは複数のオブジェクトによって表現されることを保証することによって、上記の問題点が回避される。ＭＩは、MEの特定の能力とインタフェースする１つまたは複数のメソッドを含む。従って、他の個々のオブジェクトはこれらＭＩを再使用することができる。更に、これらＭＩは再コンパイルなしに実行時に出し入れができる。
【００１８】ＭＩの細分性は、それらの有用性に影響を及ぼさない設計決定である。ＭＩは、無関係な関数からなる比較的大きいセットであろうが、あるいは密接に関係する関数からなる小さいセットであろうが、どのようなセットをも含むことができる。例えば、すべての新し

いLaserJetプリンタにおいて見出される機能のための関数を含むNewLaserJetと呼ばれるＭＩを定義することができる(LaserJetはヒューレット・パッカード社の商標である)。また、同じ関数セットを、"ChangeOnlineState"(オンライン状態変更)および"PrintTestPage"(テスト・ページ印刷)のような特定の機能に関連するいくつかのＭＩに置くことができる。複数継承を使用していくつかの細いインタフェースから粗いインタフェースを組み立てることによって、一層細かい細分性インタフェースおび一層粗い細分性インタフェースの両方をサポートすることが可能である。粗い細分性インタフェースの使用は少ないコードですむが、細かい細分性インタフェースは、典型的には、理解が一層容易であり一層大きな拡張性を提供する。

【００１９】図１は、当該ＭＥによって表される役割を提供する装置に関する機能情報の取り出しを可能にする複数ＭＩに関連づけられた典型的なＭＥオブジェクトを示している。各ＭＩは、１つまたは複数の関数を含み、それら関数の一部は、装置からの情報の取り出しを可能にし、その他の関数は、関連装置が関数を実行することを可能にする。上述のように、ＭＥは、ある装置によって実行される役割に対応し、その装置が１つの役割だけを実行しない限り必ずしも全体としてその装置に対応しない。

【００２０】図２の継承流れ図は、複数ＭＩがどのようにＭＥに関連するかを示す統一モデル化言語流れ図である。ＭＥボックスをＭＩおよびManagementInterfaceProviderボックスに接続する矢印は、継承を表す。具体的には、プログラムがコンパイルされる時、継承は、継承されるインタフェースのすべてのメソッドをＭＥオブジェクトが実施することを指示する。ＭＥオブジェクトは、ＭＩプロバイダとＭＩ自身の両方である。図２に示されているManagementInterfaceProvider(管理インタフェース・プロバイダ)オブジェクトは、ＰＳＭソフトウェアによって利用される３つの関数を含む。関数"HasManagementInterface()"は、ＭＥが指定されたＭＩにポインタを戻すことができるか否かを決定する。関数"GetManagementInterface()"は、関連ＭＥに関する情報を取り出すためまたはめに何らかの関数を実行させるために利用されるべきＭＩを戻す。関数"GetManagementInterfaceList()"は、関連するＭＩのリストを返す。

【００２１】図３は、Peripheral(周辺機器)、PeripheralServer(周辺サーバ)およびPeripheralClient(周辺クライアント)など異なる種類のＭＥを示している。それらは、プリント・パスの一部であるという共通の関係を識別するためPrintPathManagedEntity(プリント・パス管理対象エンティティ)から導出されている。図３は、完全な図であることを意図されてなく、他の種類のＭＥを含むことが可能である。例えば、プリンタは、ページを折り重ね、封筒に詰め込む第３者アドオン出力ビンを持つことができる。そのような装置は、プリンタを表示しているＭＥとは別のＭＥであることができる。ＭＩは他のＭＩから導出されることができる。導出されたＭＩがＭＥによってサポートされる時、そのＭＥは、その導出されたＭＩの親のＭＩをサポートしなければならない。多くのコンポーネント技術がインタフェース継承をサポートするので、親インタフェースの提供は通常容易である。

【００２２】図４は、典型的な管理対象エンティティ・クラスが導出される方法の１つの例を示している。図４は、ChangeOnlineStateおよびPrintTestPageという(ＭＥではない)２つの管理インタフェースを含む。LaserJet2000およびLaserJet2001は必ずしもユニークなＭＥとして定義される必要はない。それらは、単にプリンタＭＥとして定義されることができ、サポートされる他のどのようなＭＩをも提供することができる。

【００２３】ＭＥの位置

ＭＥオブジェクトは、次の３つの場所のうちの１つに配置することができる。

・それが表すマシンの範囲内に埋め込まれる(以下に例が示される)。

・それが表すマシンに直接接続されるマシンに配置される。

・管理サーバ・マシンの範囲内に配置される。

【００２４】ＭＥオブジェクトを埋め込む主な利点は、既存のプロトコルに対する依存性の排除である。ＭＥオブジェクトが装置に埋め込まれるとすれば、そのＭＥオブジェクトはその装置と装置固有の形態で対話することができる。埋め込まれたＭＥオブジェクトを持つすべての既存の周辺機器を改造することは実際的でないので、ＭＥオブジェクトを管理サーバ・マシンに配置すれば、ＭＥオブジェクトは既存のプロトコルを使用して通信することができる。これは、管理サーバ・マシンの範囲内にオブジェクトを配置する主要な利点である。直接接続されたマシンにＭＥを配置する主要な利点は、周辺機器がネットワークに接続されていない状況を克服することができる点である。分散型コンポーネント技術はネットワーク接続性を必要とするので、直接接続されたマシン上のＭＥは、ネットワーク経由のアクセスが不可能な直接接続された装置のプロキシの役割をつとめなければならない。

【００２５】ＭＥの位置は時間経過と共に変わり得る。ＩＰアドレスで構成されなかった埋め込みＭＥを持つ周辺機器を考察すると、ＩＰアドレス(または他のなんらかのネットワーク・プロトコル・サポート)がなければ、埋め込みＭＥと通信することは不可能である。この状況に対処するため、管理サーバ・マシン上にプロキシＭＥが作成される。このプロキシＭＥは、埋め込みＭＥによって提供されるすべてのインタフェースをサポートしないが、ＩＰアドレスがセットされることを可能にす

るインタフェースをサポートする。ＩＰアドレスがセットされた後、埋め込みＭＥが、管理サーバ・マシン上に配置されたＭＥプロキシを置き換える。データベースおよびエクスプローラ・アブストラクト(後述)は、ＭＥを作成し置き換える責任を持つ。

【００２６】エクスプローラ

定義

エクスプローラは、ＭＥに対応する装置を発見し、(必要に応じて)これらの装置に対応するＭＥを作成し、それらを追跡するために使用されるデータベースにＭＥ参照を送り届けるオブジェクトである。エクスプローラは、状態を持ち、構成されることができる。

【００２７】オペレーションおよびビューは、装置と直接対話せず、もう１つのオブジェクトＭＥと対話する。これは、特定の装置と通信する方法を知ることからオブジェクトを解放する。なぜなら、ＭＥアブストラクトの種々の実施形態がそのような知識をカプセル化するからである。ＭＥは、それらが表す装置に埋め込まれるか、あるいは、他のなんらかのマシン(すなわちプロキシ)に配置される。埋め込まれたＭＥは、それらが位置する装置がブートされる時、インスタンス化される。埋め込まれないＭＥは、それらが表す装置が発見されるまで、存在しない。装置が発見される時、適切なタイプのＭＥがインスタンス化される(適切なタイプは発見の間に行われる装置の照会によって決定される)。

【００２８】プリンタ役割を実行するＭＥのため１つのエクスプローラ、スキャナ役割を実行するＭＥのため１つのエクスプローラというように、異なるＭＥを発見するため異なるエクスプローラが提供される。エクスプローラは、装置を単に発見し、必要に応じてＭＥを作成し、ＭＥをデータベース・アブストラクトに渡す。ＭＥに対応する数百種類の装置があり、更に年々多数が加えられる。この状態に対処するため、エクスプローラは、特定の装置タイプに対応するＭＥを作成する方法を決定するため、装置タイプをＭＥ作成プロシージャにマップするテーブルを使用する。ＭＥは通信メカニズムをカプセル化するので、特定タイプの装置が接続される方法に従って異なるタイプのＭＥが作成される。例えば、直接接続されるレーザー・プリンタは、ネットワークに接続するレーザー・プリンタとは異なるタイプによって表されることができる。ネットワーク装置を発見する場合とは異なるタイプのエクスプローラが直接接続装置を発見するために使用されるので、そのエクスプローラは、ＭＥを構築する異なるＭＥ作成プロシージャを利用する。

【００２９】ＭＥをインスタンス化する特定の方法を上述したが、上述の方法はＭＥインスタンス化の単なる１例に過ぎない点は特に注意されるべきことであろう。各エクスプローラは、ＭＥをインスタンス化するそれ自身のメカニズムを自由に実施することができる。実際、既存のＭＥ(例えば装置の範囲内に埋め込まれたＭＥ)を発見する時、エクスプローラは、ＭＥを全くインスタンス化しない(装置がブートした時それらはすでにインスタンス化されているので)。

【００３０】エクスプローラの例：LocalIPSubnetExplorer

最大の関心を持つ装置がローカルのサブネットに配置されている装置であることが多い。これらの装置はネットワーク管理プログラムによって発見されることができるが、データベースのためのレコードの生成の代わりに、適切なタイプのＭＥが作成される。

【００３１】DirectConnectExplorer：すべてのローカル・ポートに直接接続された装置がないか検出し、検出に基づいて、直接接続された装置のプロキシのはたらきをする適切なタイプのＭＥを作成するエクスプローラが作成される。

【００３２】ServerExplorer：サーバからＭＥを"発見する"エクスプローラが作成される。日常的ルーチンとして発見を実行するサーバがすでに存在する環境においてそのようなエクスプローラが使用される。これは、ネットワーク・トラフィックを減らしてパフォーマンスを向上させるため使用されることができる。

【００３３】ManagedEntityExplorer

ネットワーク上のあらゆるＭＥを発見することは適切ではない。多くのＭＥは、管理サーバに配置されるプロキシであり、その管理サーバ上の管理アプリケーションのためだけに使用されるように意図されている。しかしながら、遠隔周辺管理ソフトウェアによって使用される意図をもって周辺機器の範囲内に埋め込まれているＭＥもある。ＰＳＭソフトウェアは、ＣＯＢＲＡ(Common Object RequestBroker Architecture)のような既存の名前付けサービスを使用して発見することが適切でないＭＥを発見することなく、発見することが適切なＭＥを発見することができる。ＭＥ自体は、それらが外部的にアクセスされるべきか否かを知っているので、名前付けサービスに名前を登録することができるので、遠隔に配置することが可能である。この解決策は、ネットワーク上に単一の名付けサービスが確立されることを必要とする。もう１つの解決策は、別のプロトコル(例えばＳＮＭＰ)を使用して装置を発見し、そのプロトコルの属性を使用して装置に埋め込まれたＭＥを識別するものである。基礎をなすプロトコルの各々が埋め込まれたＭＥの発見を助けるため異なるエクスプローラを必要とするが、名前付けサービスを備えることは避けられる。

【００３４】オペレーション

定義

オペレーションは、ＭＩを使用するＭＥの集合に作用するオブジェクトである。作用される集合における各ＭＥは、オペレーションによって必要とされるＭＩをサポートしなければならない。オペレーションは、状態を持ち、構成されることができる。

【００３５】新しいＭＩが定義されると、それらのＭＩを使用する新しいオペレーションが定義されることができる。ＰＳＭアプリケーションは、一般に、特定のオペレーションについての特定の知識なしに、オペレーションをサポートすることができる。１つのオペレーションは、それが作用するＭＥからの一定のＭＩを必要とする。１つのオペレーションは、複数のＭＩを必要とすることができるが、少なくとも１つのＭＩを必要とする。オペレーションに従って、集合における１つのＭＥの順序が有意であることもそうでないこともある。例えば、ＭＥの集合にＩＰアドレスを割り当てるオペレーションの場合、順序は重要である。しかし、ファームウェアをアップグレードするために使用されるオペレーションの場合順序は重要でない。

【００３６】オペレーションは、構成可能なものもそうでないものもあり得る。単に状態をオンラインに変更するオペレーションは、構成を全く必要としない。構成可能なオペレーションに関して、そのタイプのオペレーションを構成する方法を知っているアプレットが提供される。アプレットは、状態が修正されることを可能にするユーザ・インタフェースを提供する責任を持つ。そのようなアプレットは、管理クライアント・マシン上で実行され、メソッドを起動することによってオペレーションを構成する。１つの好ましい実施形態において、アプレットはJavaで書かれる。

【００３７】例

オンライン・オペレーション

図４のクラス階層に示されているChangeOnlineStateというＭＩを参照すれば、このＭＩは、このＭＩをサポートしているＭＥのオンライン状態が変更されることを可能にする。オンライン状態がオンラインであることを保証するため、"ChangeOnlineState"ＭＩを使用するオペレーションが構築される。オンライン・オペレーションは、すべてのプリンタを定期的にオンラインに復元する管理アプリケーションを構築するため使用することができる。

【００３８】InstallFontオペレーション

ツルー・タイプ・フォントを管理するために使用される次のようなインタフェースを考察する。

【００３９】

【表１】

```
interface TrueTypeFontManagement:
ManagementInterface
{
  void Add( Font f);
  void FontList List();

  void Remove(FontListElement f);
};
```

【００４０】ここで、エンドユーザ・マシン(例えばPC)およびプリンタ上にフォントを導入する周辺システム管理アプリケーションを考察する。そのようなアプリケーションは、TrueTypeFontManagementインタフェースをサポートするマシン上にフォントを導入するオペレーションを使用して構築されることができる。アプリケーションは、単にTrueTypeFontManagementインタフェースをサポートするすべてのＭＥを識別し、所望のフォント・ファイルに関するInstallFontオペレーションを構成し、識別されたＭＥに対するオペレーションを起動する。

【００４１】AssignSequentialNetworkAddressesオペレーション

１つのサブネットから他のサブネットへ物理ネットワークを変更する時、物理ネットワーク上のすべての装置の番号を再割り当てする必要がある。ネットワーク管理プログラムは、物理ネットワーク上の(そのようなアプリケーションによって管理されている)すべての装置にネットワーク・アドレスをシーケンシャルに再割り当てするメカニズムをサポートする。これは、オペレーションを使用して達成される。そのようなオペレーションはネットワーク・アドレスの範囲を用いて構成される。次に、番号を再割り当てされるべき周辺機器が、(アプリケーション次第でユーザの介入があることもないこともあるが)アプリケーションによって選択される。最後に、選択された周辺機器ＭＥの集合に対して、AssignSequentialNetworkAddressesオペレーションが起動される。

【００４２】ビュー

定義

ビューは、ＭＩを使用するＭＥの集合を表示するオブジェクトである。表示されている集合における各ＭＥは、ビューによって必要とされるＭＩをサポートしなければならない。ビューは、状態を持ち、構成されることができる。ビューは、オペレーションとほとんど同じであるが、オペレーションが与えられたＭＥに対して作用するのに対して、ビューはＭＥを表示する点が相違する。ビューは、状態、構成および表示という３つの部分に分解される。状態は、現在の構成を追跡し、ユーザ・インタフェースを持たない。構成は、オペレーションのための構成アプレットと同様である。構成は、状態が修正されることを可能にするユーザ・インタフェースを提示する責任を持つ。表示は、ＭＥの集合を表示する責任を持つ。ＭＥの順序は、ビューによっては、重要であることも、そうでもないこともある。一部のビューは、集合におけるＭＥの順序を無視して、単に上述の３つの関係ＭＥを表示する(残りはまとめられる)。別のビューは、集合に与えられる順序ですべてのＭＥを表示する。異なる状態を持つビューの複数のインスタンスが同時に存在することができる。

【００４３】例

消耗品ビュー
プリンタがトナーを使い果たす時手間がかかる。 ユーザが訓練されていないことが多いので、トナー・カートリッジの交換は、技術者がカートリッジを取り替える場合より長い時間を要する。また、訓練されていないユーザは、プリンタを偶然傷つけることがある。この問題に対処するため、使い果たす前にトナーを交換することができるように、どのプリンタがトナーを使い果たしそうか検出する手段を持つことは役立つ。これは、ビューを使用して達成されることができる。そのようなビューは、一定のトナーしきい値より下のプリンタを、トナーの量の少ないものから多いものへの順番で、表示する。トナーしきい値を変更することによって、ユーザは表示されるプリンタの数を変えることができる。トナーしきい値が１００%にセットされると、すべてのプリンタが表示される。

【００４４】スキャナ・ビュー
このビューは、特定のタイプのＭＥを表示する。プリンタに関してビューを作成することができる。このビューは、プリンタが印刷する分あたりの最大ページ数を示す。スキャナの場合、このビューは、スキャナがカラーをサポートするか否かを標示するフィールドを持つ。このように、特定タイプのＭＥをサポートしそのタイプにとって意味のあるフィールドを表示するビューを作成することができる。

【００４５】マップ・ビュー
マップで構成され、マップ上の位置に従ってＭＥを表示するビューを構築することができる。ＭＥの位置は、マップ・ビューの状態を含む位置ファイルから導出される。ビューが状態を持つので、ユーザとビューの間の交信から派生する情報を記憶するためその状態を使用することができる。

【００４６】データベース
定義
データベースは、ＭＥを受け入れ、複製を取り扱い、後の使用に備えてＭＥを記憶するオブジェクトである。また、特定のＭＥまたはすべてのＭＥについてデータベースを消去することができる。

【００４７】データベース・アブストラクトは、(エクスプローラのような)他のアーキテクチャ上のアブストラクトが特定タイプのデータベースについての知識を持つことなく多くのタイプのデータベースと対話することを可能にする。使用中のデータベースの特定のタイプについて知識を持つアプリケーションは、当然のことながら、どのようなデータベース管理システムが基礎として使用されていようともそれらの全機能性を自由に使用することができる。データベース・アブストラクトの意図は、エクスプローラが対話することができ、アプリケーションの記憶メカニズムとの通信を可能にするオブジェクトを提供することにある。

【００４８】例
ファイル型データベース
アプリケーションによっては、単にＭＥのリストを含むファイルを持つことを受け入れる。ＭＥに関連した情報がファイルに記録されるように、データベース・アブストラクトに関して指定されるインタフェース(例えばadd、removeOne、removeAll)を実施することによってファイル型データベースを作成することができる。ＭＥがデータベースに加えられる時、アプリケーションによって必要とされる情報と共に、(removeOneおよび複製処理をサポートするため)ＭＥをユニークに識別するために必要とされる情報を含む行がファイルに付加される。

【００４９】オブジェクト指向データベース
一部のアプリケーションの場合、オブジェクト指向データベースによって提供される柔軟性が魅力的である。そのようなデータベースは、ファイル型データベースの場合と同様に(但しオブジェクト指向データベースに基づいて)実施されることができる。

【００５０】リレーショナル・データベース
高いデータベース・パフォーマンスを必要とするアプリケーションの場合、リレーショナル・データベースを使用することが魅力的かもしれない。そのようなデータベースは、ファイル型データベースの場合と同様に(但しリレーショナル・データベースに基づいて)実施されることができる。

【００５１】システム構成
図５には、本発明を実施するシステムが図示されている。クライアント・コンピュータ１０は、中央処理装置(ＣＰＵ)１２、ネットワーク相互接続モジュール１４、およびＰＳＭオブジェクト１８を保有するメモリ１６を含む。ＰＳＭオブジェクト１８は、ネットワーク２０を経由してアクセス可能な複数のPeripheral(周辺装置)を管理するため、上述の種々のオブジェクト(Object)を利用するために必要なコードを含む。

【００５２】周辺装置２２および２４の各々は、ManagementInterfaceProvider(管理インタフェース・プロバイダ)オブジェクト２６を含む。ManagementInterfaceProviderオブジェクト２６は、"has interface method"メソッド８、"list interface method"メソッド３０および"get interface method"メソッド３２へのアクセスを可能にする。各周辺装置は、更に、１つまたは複数の埋め込みＭＥオブジェクトを含み、それら埋め込みＭＥ(Managed Entity)オブジェクトの各々は、それぞれの周辺装置役割に関する情報の検索を可能にする１つまたは複数のＭＩ(Management Interface)への参照を可能にする。上述のように、各ＭＩは、その実行がそれぞれの周辺装置によって演じられる役割に関する情報の検索を可能にする少なくとも１つの関数を含むオブジェクトである。

【００５３】以下の記述においては、すべてのオブジェクトがすでにメモリにロードされ、使用の準備ができて

いると仮定する。しかしながら、磁気ディスク２９のような１つまたは複数のデータ記憶装置上にそのようなオブジェクトを記憶しておき、その内容を必要に応じてメモリにロードすることも可能である。

【００５４】メモリ１６は、更に、選択されたＭＥ、状態および構成の表示を可能にするビュー・オブジェクト３６、ＭＩを使用するＭＥの集合に作用するオペレーション・オブジェクト３８、データベース・オブジェクト４０というＭＥオブジェクトに対する参照を取得するためのMethod(メソッド)を各々含む１つまたは複数のエクスプローラ・オブジェクト３４を含む。

【００５５】例としてのアプリケーション："UpdateFirmware"

図６および図７を参照して、ＰＳＭプロシージャ１８のオペレーションの詳細を以下に記述する。図６および図７は、ネットワーク・プリンタのファームウェアをアップグレードする責任を持つアプリケーションを例示する本発明の方法の論理の流れ図を含む。このアプリケーションは、次のような活動を実行する。

１．ネットワーク上のすべての使用可能プリンタを特定する。

２．それぞれのファームウェアをアップグレードおよび構成する能力を備えるプリンタのプリンタ・モデル別リストを表示する。

３．プリンタ・リストから１つまたは複数のプリンタをユーザが選択することを可能にする。

４．選択されたプリンタの各々のファームウェアをアップグレードする。

【００５６】背景

プリンタは、プリント・ジョブを処理するために使用される(ＲＯＭまたはフラッシュメモリのような他のなんらかの半永久的メモリに記憶されるので)ファームウェアと呼ばれるソフトウェアを含む。ＰＣ上のソフトウェアと同様に、ファームウェアは、時々、欠陥を修復したり、新機能を追加する。独立した活動として各プリンタを更新するのではなく、管理者が同じモデルのすべてのプリンタのファームウェアを更新する手段を持つことが望ましい。

【００５７】実行

１．UpdateFirmware(ファームウェア更新)アプリケーションが始動する(図６ステップ１００)。この時点ではアプリケーションはＭＥをなにも持っていない。

２．アプリケーションは、１つまたは複数のエクスプローラを使用するＭＥを特定する。アプリケーションは、起動時に構築される既知のすべてのエクスプローラを含むリストを所有している。このリストはいくつかの方法で構築することができるが、単純化のため、リストは、ファイルを読み取ることによって取得されると仮定する。また、各エクスプローラは、独立した動的リンク・ライブラリ(すなわちＤＬＬ)に含まれていて、上記リストが、ＤＬＬのファイル・システムにおいて各エクスプローラ・オブジェクトのための実行可能コードを含む位置を識別すると仮定する。更に、エクスプローラのリストはただ１つのエントリを持つと仮定する。アプリケーションは、リストを読み、(オペレーティング・システム呼び出しを使用して)ＤＬＬをロードし、エクスプローラ・オブジェクトのインスタンスをインスタンス化するためＤＬＬのエクスポートされた関数を起動する(ステップ１０２)。

【００５８】３．今や、アプリケーションはエクスプローラ・オブジェクトを持つ。次のステップは、エクスプローラ・オブジェクトを使用して、ＭＥのリストを取得することであるが、それはエクスプローラ・オブジェクトの起動を必要とする。エクスプローラ・オブジェクトのメソッドは、発見されたＭＥのリストを提供するアレイのようなデータ構造を戻す。以下のＣ＋＋アブストラクト基本クラスは、エクスプローラのための可能なインタフェース(および特に、本例において使用されるインタフェース)を例示する。

【００５９】

【表２】

```
class Explorer {
  virtual void explore(ManagedEntity *MEList,
                       int           *MEListLength) = 0;
};
```

【００６０】４．アブストラクト基本クラスはエクスプローラのインタフェースを与えるが、実施形態を記述しない。多数の実施形態があり得るが、単純化のため、このエクスプローラはプリンタの範囲内に埋め込まれるＭＥのリストを含むテキスト・ファイルへの参照を含む(これらのＭＥの各々はプリンタが始動した時作成された)と仮定する。

５．エクスプローラ・オブジェクト上のエクスプローラ・メソッドを起動した後、アプリケーションはアレイの形態のＭＥの集合を持つ。次のステップは、RemoteFirmwareUpdate(遠隔ファームウェア更新)ＭＩを供給するＭＥを表示する(ステップ１０４)。このステップを達成するため、アプリケーションは、各ＭＥがRemoteFirmwareUpdateＭＩを提供するか否か判断しなければならない。これは、ＭＥの各々の上の"hasInteface"メソッド２８を起動することによって行われる。例えば、ＭＩのインタフェースは、次のようなＣ＋＋アブストラクト基本クラスによって表されることができる。

【００６１】

【表３】

```
class ManagedEntity {
     boolean hasManagementInterface(string mi_name) = 0;
     ManagementInterface *getManagementInterface(string
 mi_name) = 0;
    void getManagementInterfaceList(string *strList,
                                    int    *strListLen) = 0;
  };
```

【００６２】６．hasInterfaceが起動される時、各ＭＥは真または偽を返さなければならない(ステップ１０６)。各ＭＥが提供する応答は、ＭＥによってサポートされているＭＩ名の各々をリストする内部テーブルを調べることによって決定される。hasInterface関数は、供給されたＭＩ名をテーブルに含まれるＭＩ名の各々と比較する。hasInterfaceは、一致が見つからなければ、偽を返し、一致が見つかれば、真を返す。上記説明は管理インタフェース名のテーブルを含むものとしてＭＥを示しているが、必ずしもすべてのＭＥがそのように実施されるとは限らない。

【００６３】７．ＭＥの各々がRemoteFirmwareUpgrade管理インタフェースをサポートしているか否か照会された後、次のステップは、このインタフェースをサポートするＭＥを表示するものである。アプリケーションはこれらのＭＥに関する情報を直接表示することはできるが、ビュー・オブジェクトを使用してこれらのＭＥを表示する方が一層柔軟である。ビュー・オブジェクトは、再使用可能であり、ここに取り上げているアプリケーション以外のアプリケーションにおいても使用することができる。ビュー・オブジェクトは、作業すべき１つまたは複数の特定ＭＩを必要とする。この場合、ビュー・オブジェクトは、RemoteFirmwareUpgradeＭＩを必要とする。RemoteFirmwareUpgrade管理インタフェースは、次のようなアブストラクト基本クラスによって表されることができる。

【００６４】

【表４】

```
class RemoteFirmwareUpgrade : ManagementInterface {
     unsigned long firmwareVersion version () = 0;
     void upgradeFirmware ( string newFirmwareURL ) = 0;
     string printerModel() = 0;
};
```

【００６５】実際の実施形態においては、printerModel関数は、なんらかの別のもっと一般的ＭＩの一部であろうが、この例では単純化のためRemoteFirmwareUpgradeＭＩに含まれている。RemoteFirmwareUpgradeＭＩは、本例を完了するために必要とされるすべての関数を含む。RemoteFirmwareUpgradeＭＩをサポートするＭＥを表示する前に、ビュー・オブジェクトをインスタンス化する必要がある(ステップ１０８)。

【００６６】８．(本例における)ビュー・オブジェクトのインスタンス化は、オペレーティング・システムＡＰＩを使用してＤＬＬをメモリにロードして、ビュー・オブジェクトを作成するメソッドを呼び出すことによって実行される。

９．一旦ビュー・オブジェクトが作成されたなら、それを使用してＭＥを表示することができる。ビューは、Ｃ＋＋アブストラクト基本クラスとして定義される次のようなインタフェースを持つものとみなされる。

【００６７】

【表５】

```
class ManagementView {
    void display(ManagedEntity *me_list,
                 int           *me_listLen) = 0;
    void getSelectedMEs (ManagedEntity *me_list,
                         int           *me_listLen) = 0;
};
```

【００６８】ＭＥの集合を表示するため、ＭＥポインタのアレを用いるdisplay(表示)メソッドが起動される。それに応じて、ビュー・オブジェクトは以下のようにオペレーションを実行する。

・各ＭＥに関してhasInterfaceメソッドを起動することによって、ＭＥリスト・アレイにおいて与えられた各ＭＥがRemoteFirmwareUpgradeＭＩをサポートすることを検証する。これに応答して、前述のように、各ＭＥは内部データ構造を調べて与えられたＭＩがサポートされているか否か判断する。RemoteFirmwareUpgradeＭＩをサポートしないＭＥがＭＥリスト・アレイの中に存在すれば、エラーが発生する。

・各ＭＥから実際のRemoteFirmwareUpgradeＭＩを取得する(ステップ１０８)。これは、各ＭＥに関して所望のＭＩ名(このケースでは"RemoteFirmwareUpgrade")を用いてgetManagementInterface関数を起動することによっ

て達成される。各ＭＩは、ＭＩクラスへのポインタとして表現される。前述のように、実際のＭＥはプリンタの範囲内で遠隔場所に存在する。これは、ＭＩを構成する関数の実施もまた遠隔に存在することを意味する。ビュー・オブジェクトのようなクライアント・ソフトウェアがローカルのインタフェース(例えばRemoteFirmwareUpgrade)と対話するとしても、これらローカル・インタフェースは、プリンタの範囲内に存在する実際の実施形態からのパラメータおよびメソッドを利用する確立された通信プロシージャを使用する。一旦ＭＩが取得されると、ＭＩはRemoteFirmwareUpgradeとしてのタイプを与えられ、テーブルに記憶される。このテーブルは、ＭＥへのローカル・ポインタ、対応するＭＥに関するRemoteFirmwareUpgradeＭＩへのローカル・ポインタ、バージョン番号(後述)およびモデル(後述)を含む。

【００６９】・RemoteFirmwareUpgradeＭＩによって提供されるfirmwareVersionメソッドを起動することによって各ＭＥのバージョン番号を取得する(各ＭＥは、このＭＩのそれ自身の独立した実施形態を持つ)。取得された値は、各ＭＥに関する情報(例えばインタフェース・ポインタ)を記憶するために使用されるテーブルに記憶される。

・RemoteFirmwareUpgradeＭＩによって提供されるprinterModel関数を起動することによって各ＭＥのプリンタ・モデルを取得する(各ＭＥは、このＭＩのそれ自身の独立した実施形態を持つ)。取得された値は、各ＭＥに関する情報(例えばインタフェース・ポインタ)を記憶するために使用されるテーブルに記憶される。

【００７０】１０．今や、各ＭＥに関する情報を記憶するために使用されるテーブルが利用可能であるので、テーブルをプリンタ・モデルおよびファームウェア・バージョンによってソートすることができる。ソートは、確立されているいくつかの手段のいずれによってでも実施することができる。次に、ビュー・オブジェクトは、ウィンドウをオープンし、記憶されているＭＥ情報リストを表示することができる。このリストにおいて、ビュー・オブジェクトは、プリンタ・モデルおよびファームウェア・バージョンを特に示すことができる。ビュー・ウィンドウがクリックされる時、ビュー・オブジェクトは、テーブルにおけるどのＭＥエントリがクリックされたかを判断して、次の２つのオペレーションを行う。第１に、ビュー・オブジェクトは、ウィンドウにおいてそのエントリをハイライトする。第２に、ビュー・オブジェクトは、ＭＥテーブルにおけるフィールドを修正してそのフィールドが選択されていることを示す。

【００７１】上述のように、ビュー・オブジェクトの範囲内でＭＥを追跡するために使用されるテーブルは、ＭＥへのローカル・ポインタ、対応するＭＥに関するRemoteFirmwareUpgradeＭＩへのローカル・ポインタ、ファームウェア・バージョン番号およびモデル名を含むものとして記述された。このテーブルは、ユーザ・インタフェースを通して特定のエントリが選択されたことを標示する新しいブーリアン・フィールドを含むように拡張される。

【００７２】１１．一旦ユーザがアップグレードすべき所望のＭＥ(例えばプリンタ)を選択したならば、ユーザは、アプリケーションによって提供されるメニュー項目のようなユーザ・インタフェースを通して、ファームウェア・アップグレードが実行されるべきことを標示することができる。これは、オペレーションと呼ばれるメニューをプルダウンして、"upgrade remote firmware"を選択することによって達成されることができる。オペレーションは、ＭＥの集合に対して実行されるAction(アクション)である。しかし、オペレーションが使用されることができる前に、次の２点が起きなければならない。第１に、オペレーションがインスタンス化されなければならない。第２に、オペレーションが構成されなければならない。次のようなＣ＋＋アブストラクト基本クラスがオペレーションに関する可能なインタフェースを提供する。

【００７３】

【表６】

```
class ManagementOperation {
    void configure() = 0;
    void operate(ManagedEntity *me_list,
                 int           *me_listLen) = 0;
};
```

【００７４】１２．オペレーションの作成は、ビューおよびエクスプローラの作成の場合と同様な方法で行われる。すなわち、オペレーティング・システム呼び出しを使用してＤＬＬをメモリへロードして、そのＤＬＬによってエクスポートされた関数を起動することによって、オペレーションがインスタンス化される。

【００７５】１３．一旦作成されたならば、オペレーションは構成される必要がある。これは、オペレーション・オブジェクトによって供給される"configure"メソッドを起動することによって、実行される。このメソッドは、ユーザがオペレーション・オブジェクトを構成することを可能にする構成ダイアログを表示する。ユーザ・インタフェースは、テキスト編集ボックスおよびそれに続くラベルを含む。ラベルは、アップグレードされるべきプリンタ・ファームウェアが記憶されているターゲットＵＲＬをテキスト編集ボックスが含むべきものであることを標示する。ユーザは、プリンタ・ファームウェアが置かれている有効なＵＲＬを与えるストリングを入力する。ユーザがＵＲＬに入力したならば、オペレーション・オブジェクトは構成され、動作の準備ができる。

【００７６】１４．最初に、アプリケーションは、選択されたＭＥをオペレーション・オブジェクトに渡すことができるように、選択されたＭＥのリストを取得しなけ

ればならない(ステップ１１２)。このため、アプリケーションはビュー・オブジェクトからgetSelectedMEsメソッドを起動する。このメソッドは、ＭＥのテーブルをアクセスして、選択されている各ＭＥをアレイに書き込む。その後、このアレイはアプリケーションに返される。今や、アプリケーションはアレイの中に選択されたＭＥを持つので、オペレーション・オブジェクトのメソッドを起動することができる。

【００７７】１５．メソッドは次のように動作する。

・各ＭＥのhasInterfaceを起動することによって、渡されたＭＥの各々がRemoteFirmwareUpgradeＭＩを所有していることを検証する。

・各ＭＥに関してgetInterfaceメソッドを起動することによってRemoteFirmwareUpgradeＭＩに対するローカル・ポインタを取得する。これは、RemoteFirmwareUpgradeＭＩのＭＩ名をgetInterfaceメソッドに渡すことによって実行される。供給されたＭＥのすべてがRemoteFirmwareUpgradeＭＩを提供することを確認した後、各ＭＥはＭＥごとにそれぞれ個別に処理される。換言すれば、次のステップが完了した後、ＭＥの処理が開始する。

・RemoteFirmwareUpgradeＭＩを取得した後upgradeFirmwareメソッドを起動する。upgradeFirmwareメソッドはRemoteFirmwareUpgradeＭＩの一部として提供される。このメソッドは、ファームウェアが置かれているＵＲＬを記述するストリングを受け入れる。

【００７８】上述のように、ＭＥおよびＭＩは、ローカル・プロキシ・オブジェクト(すなわち本例によって使用されているアブストラクト基本クラス)によって表されている。これらのクラスは、通信プロトコルのためフロントエンドを提供する。実際に実施されるupgradeFirmwareメソッドは、各プリンタに配置される(上述のように各プリンタは埋め込みＭＥを含む)。upgradeFirmwareメソッドが起動される時、各プリンタの範囲内に実施されるメソッドが、ＵＲＬによって指定されるサーバからファームウェアをダウンロードするために必要なActionを実行する。一旦ファームウェア・イメージがダウンロードされると、プリンタは、チェックサムの確認(または他の適切な技法)によってファームウェア・イメージを検査し、ファームウェア・イメージを半永久記憶装置へ書き込む。最後に、プリンタは、新しいファームウェアを実行させるためそれ自身をリセット(ウォーム・リブート)する。各ＭＥが処理され、すべてのファームウェア・イメージがアップグレードされたならば、アプリケーションは終了する。

【００７９】上記記述が本発明を例示する目的からのみ行われた点は理解されるべきことであろう。本発明の理念を逸脱することなく当業者によって上記実施形態に種々の修正および変更を加えられることができるであろう。

【００８０】本発明には、例として次のような実施様態が含まれる。

(1)管理コンピュータが周辺システムの装置を管理することを可能にするシステムであって、管理対象装置の各々が、必ずしも上記管理コンピュータではないコンピュータのため１つまたは複数のサービスを提供すると共に、インタフェース・オブジェクトに関するメソッドが実行されることを可能にする管理インタフェース・プロバイダ・オブジェクトを記憶し、上記サービスの各々が、１つまたは複数の管理インタフェースに対する参照を含む関連する管理対象エンティティ・インタフェースを有し、上記管理インタフェースの各々が、上記管理対象エンティティによって表され、上記管理対象装置によって実行される上記サービスの機能に関する情報を提供するかまたは上記管理対象装置が関数を実行することを可能にするか少なくともいずれかを実行する１つまたは複数のメソッドを備え、上記管理コンピュータが、上記管理対象装置の各々との通信を可能にする入出力手段と、エクスプローラ・オブジェクトを記憶するメモリと、管理対象装置のサービスに関するユーザ照会に応答して、上記管理対象装置のサービスに対応する管理対象エンティティを決定するため上記エクスプローラ・オブジェクトを起動し、また、上記照会に関連し上記管理対象装置に関する情報へのアクセスを可能にするかまたは上記管理対象装置が上記照会に関連する関数を実行することを可能にする上記管理対象エンティティに関連する管理インタフェースを復元するため上記インタフェース・プロバイダ・オブジェクトを起動するプロセッサ手段と、を備える、周辺装置管理システム。

【００８１】(2)上記メモリがビュー・オブジェクトを含み、上記プロセッサが、上記ビュー・オブジェクトに関連するメソッドを起動して上記管理対象装置サービスに対応する上記管理対象エンティティのうちの少なくとも１つを表示し、それによって、ユーザが上記インタフェース・プロバイダ・オブジェクトの関数を起動する前に１つまたは複数の上記管理対象エンティティを選択することを可能にする、前記(1)に記載の周辺装置管理システム。

(3)上記メモリがビュー・オブジェクトを含み、上記プロセッサが、上記ビュー・オブジェクトに関連するメソッドを起動して上記管理対象エンティティのグループの状態を表示する、前記(1)に記載の周辺装置管理システム。

(4)上記管理インタフェース・プロバイダ・オブジェクトが、照会された機能に対応する関連する管理インタフェースを有するか否かを判断するメソッドと、上記関連する管理インタフェース・オブジェクトを取り出すメソッドと、管理対象エンティティに関連する管理インタフェース・オブジェクトをリストするメソッドと、を含み、上記プロセッサが、上記管理対象エンティティに関連する上記管理インタフェース・オブジェクトを復元す

るため上記メソッドの少なくとも１つを使用する、前記(1)に記載の周辺装置管理システム。

【００８２】(5)特定のサービスを含む上記管理対象装置の各々が、実質的に同等の管理対象エンティティ・インタフェースで上記サービスを表す、前記(1)に記載の周辺装置管理システム。

(6)特定の機能を含む上記管理対象装置の各々が、実質的に同等の管理対象インタフェース・オブジェクトで上記機能を表す、前記(1)に記載の周辺装置管理システム。

(7)上記管理対象装置の少なくとも一部が１つのネットワークを経由して上記コンピュータと相互接続される、前記(1)に記載の周辺装置管理システム。

【００８３】(8)周辺システムの装置を管理するため管理コンピュータを制御するメモリ媒体であって、管理対象装置の各々が、必ずしも上記管理コンピュータではないコンピュータのため１つまたは複数のサービスを提供すると共に、インタフェース・オブジェクトに関するメソッドが実行されることを可能にする管理インタフェース・プロバイダ・オブジェクトを記憶し、上記サービスの各々が、１つまたは複数の管理インタフェースに対する参照を含む関連する管理対象エンティティ・インタフェースを有し、上記管理インタフェースの各々が、上記管理対象エンティティによって表され、上記管理対象装置によって実行される上記サービスの機能に関する情報を提供するかまたは上記管理対象装置が関数を実行することを可能にするか少なくともいずれかを実行する１つまたは複数のメソッドを備え、上記管理コンピュータが上記管理対象装置の各々との通信を可能にする入出力手段およびエクスプローラ・オブジェクトを記憶するメモリを有し、管理対象装置のサービスに関するユーザ照会に応答して、上記管理対象装置のサービスに対応する管理対象エンティティを決定するため上記エクスプローラ・オブジェクトを起動するように上記管理コンピュータを制御する手段と、上記照会に関連しかつ上記管理対象装置に関する情報へのアクセスを可能にするか、または、上記管理対象装置が上記照会に関連する関数を実行することを可能にする上記管理対象エンティティに関連する管理インタフェースを復元するため上記インタフェース・プロバイダ・オブジェクトを起動するように上記管理対象装置を制御する手段と、を備える、メモリ手段。

【００８４】(9)上記メモリがビュー・オブジェクトを含み、上記ビュー・オブジェクトに関連するメソッドを起動して上記管理対象装置サービスに対応する上記管理対象エンティティのうちの少なくとも１つを表示し、それによって、ユーザが上記インタフェース・プロバイダ・オブジェクトの関数を起動する前に１つまたは複数の上記管理対象エンティティを選択することを可能にするように上記管理コンピュータを制御する、前記(8)に記載のメモリ手段。

(10)上記メモリがビュー・オブジェクトを含み、上記ビュー・オブジェクトに関連するメソッドを起動して上記管理対象エンティティのグループの状態を表示するように、上記管理コンピュータを制御する、前記(8)に記載のメモリ手段。

【００８５】(11)上記管理インタフェース・プロバイダ・オブジェクトが、照会された機能に対応する関連する管理インタフェースを有するか否かを判断するメソッドと、上記関連する管理インタフェース・オブジェクトを取り出すメソッドと、管理対象エンティティに関連する管理インタフェース・オブジェクトをリストするメソッドと、を含み、上記管理対象エンティティに関連する上記管理インタフェース・オブジェクトを復元するため上記メソッドの少なくとも１つを使用するように上記管理コンピュータを制御する、前記(8)に記載のメモリ手段。

(12)特定のサービスを含む上記管理対象装置の各々が、実質的に同等の管理対象エンティティ・データ構造で上記サービスを表す、前記(12)に記載のメモリ媒体。

(13)特定の機能を含む上記管理対象装置の各々が、実質的に同等の管理対象インタフェース・オブジェクトおよび内包されたメソッドで上記機能を表す、前記(12)に記載のメモリ媒体。

【００８６】

【発明の効果】本発明によると、オブジェクト・インタフェースを利用して、効率的で、拡張性の高い周辺システム管理が実現される。

【図面の簡単な説明】

【図１】典型的なＭＥデータ構造のブロック図である。

【図２】ＭＥがＭＩのサブクラスであるという関係を示す統一モデル言語表現を表すブロック図である。

【図３】種々の種類のＭＥを示すブロック図である。

【図４】ＭＥクラスの導出の１例を示すブロック図である。

【図５】本発明の方法を実行するシステムの構成を示すブロック図である。

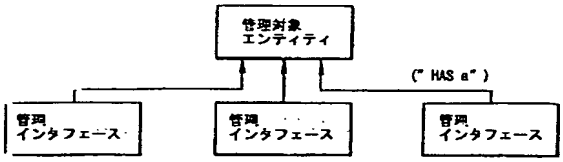
【図６】図７と共に、本発明の方法の１例の論理の流れ図である。

【図７】図６と共に、本発明の方法の１例の論理の流れ図である。

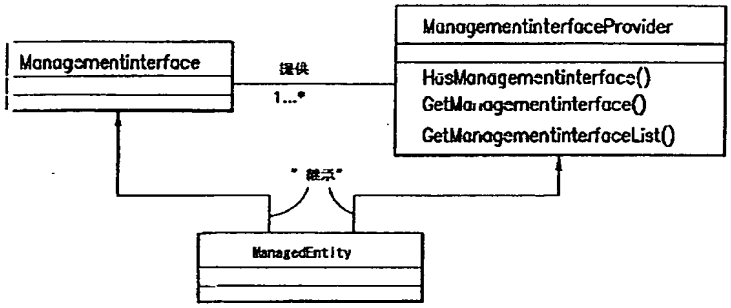
【符号の説明】

１０	管理コンピュータ
１４	ネットワーク相互接続機構すなわち入出力手段
１６	メモリ
２２、２４	管理対象周辺装置
２６	管理インタフェース・プロバイダ・オブジェクト
３３	管理対象エンティティ

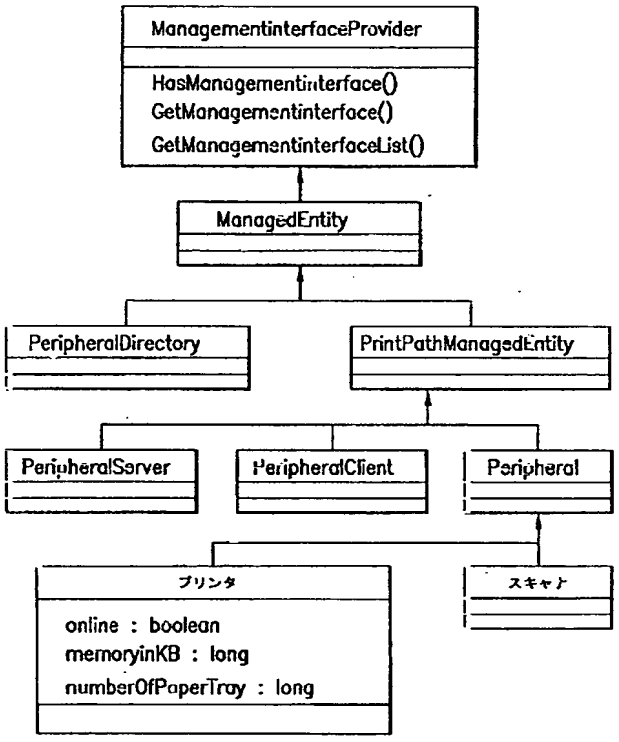
【図1】


管理対象
エンティティ
("HAS a")
管理
インタフェース
管理
インタフェース
管理
インタフェース

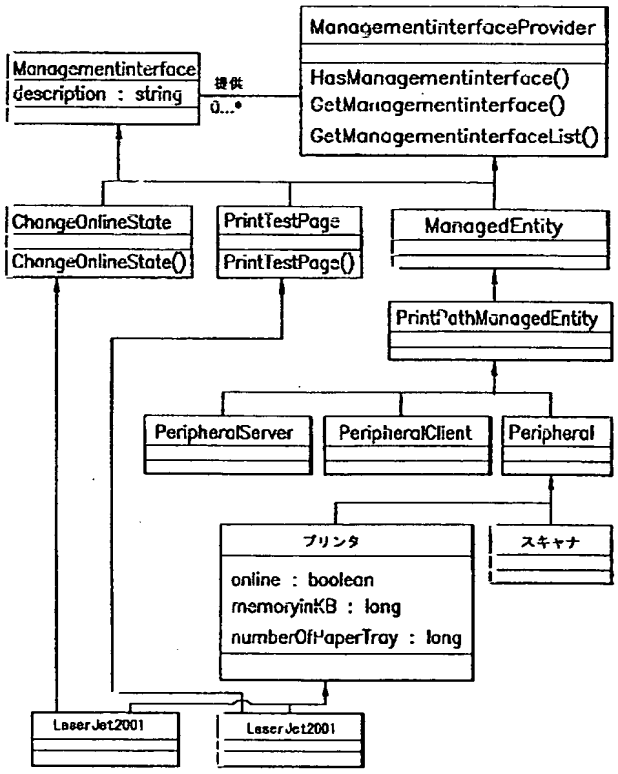
【図2】


ManagementinterfaceProvider
Managementinterface
提供
1...*
HasManagementinterface()
GetManagementinterface()
GetManagementinterfaceList()
"継承"
ManagedEntity

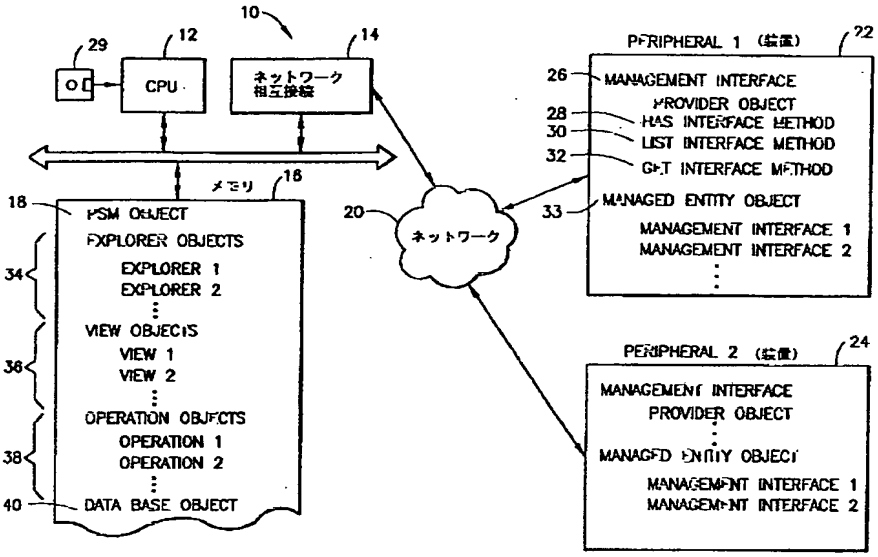
【図3】


ManagementinterfaceProvider
HasManagementinterface()
GetManagementinterface()
GetManagementinterfaceList()
ManagedEntity
PeripheralDirectory
PrintPathManagedEntity
PeripheralServer
PeripheralClient
Peripheral
プリンタ
online : boolean
memoryinKB : long
numberOfPaperTray : long
スキャナ

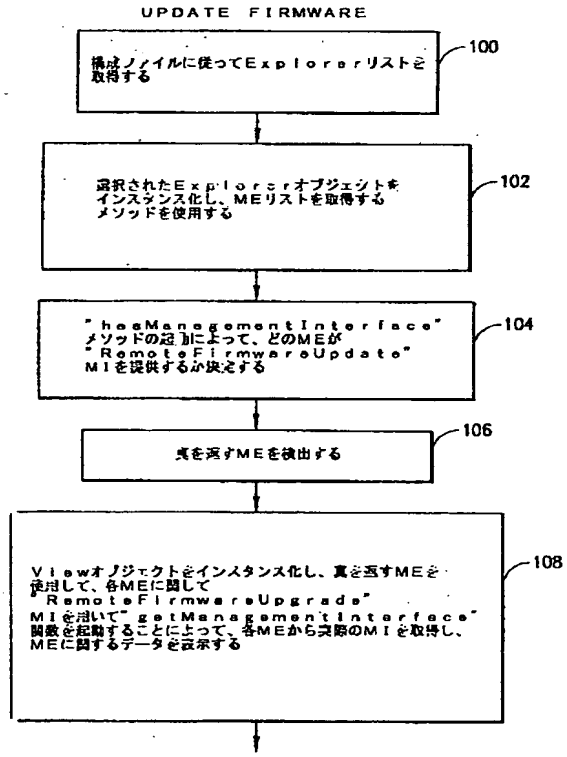
【図4】


ManagementinterfaceProvider
Managementinterface
description : string
提供
0...*
HasManagementinterface()
GetManagementinterface()
GetManagementinterfaceList()
ChangeOnlineState
ChangeOnlineState()
PrintTestPage
PrintTestPage()
ManagedEntity
PrintPathManagedEntity
PeripheralServer
PeripheralClient
Peripheral
プリンタ
online : boolean
memoryinKB : long
numberOfPaperTray : long
スキャナ
LaserJet2001
LaserJet2001

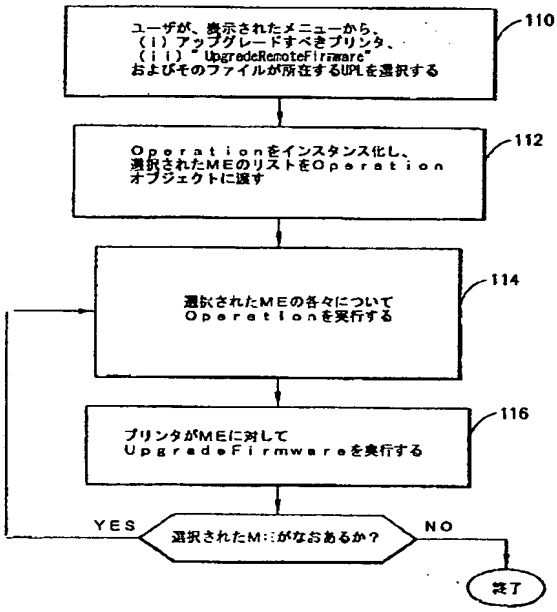
【図5】


10
29
12
14
CPU
ネットワーク
相互接続
メモリ
18
PSM OBJECT
EXPLORER OBJECTS
EXPLORER 1
EXPLORER 2
VIEW OBJECTS
VIEW 1
VIEW 2
OPERATION OBJECTS
OPERATION 1
OPERATION 2
DATA BASE OBJECT
34
36
38
40
20
ネットワーク
PERIPHERAL 1 (装置)
22
26
MANAGEMENT INTERFACE
PROVIDER OBJECT
28
HAS INTERFACE METHOD
30
LIST INTERFACE METHOD
32
GET INTERFACE METHOD
33
MANAGED ENTITY OBJECT
MANAGEMENT INTERFACE 1
MANAGEMENT INTERFACE 2
PERIPHERAL 2 (装置)
24
MANAGEMENT INTERFACE
PROVIDER OBJECT
MANAGED ENTITY OBJECT
MANAGEMENT INTERFACE 1
MANAGEMENT INTERFACE 2

【図6】


UPDATE FIRMWARE
構成ファイルに従ってExplorerリストを取得する
100
選択されたExplorerオブジェクトをインスタンス化し、MEリストを取得するメソッドを使用する
102
"hasManagementInterface"メソッドの返り値によって、どのMEが"RemoteFirmwareUpdate"MIを提供するか決定する
104
真を返すMEを検出する
106
Viewオブジェクトをインスタンス化し、真を返すMEを使用して、各MEに関して"RemoteFirmwareUpgrade"MIを用いて"getManagementInterface"関数を起動することによって、各MEから実際のMIを取得し、MEに関するデータを表示する
108

【図7】


ユーザが、表示されたメニューから、
(i) アップグレードすべきプリンタ、
(ii) "UpgradeRemoteFirmware"
およびそのファイルが所在するURLを選択する
110
Operationをインスタンス化し、選択されたMEのリストをOperationオブジェクトに渡す
112
選択されたMEの各々についてOperationを実行する
114
プリンタがMEに対してUpgradeFirmwareを実行する
116
YES
選択されたMIがなおあるか?
NO
終了

フロントページの続き

(72)発明者　ジェームズ・クラフ
アメリカ合衆国83642アイダホ州メリディアン、イースト・ワクリー・ストリート465

(72)発明者　エム・スコット・ガートナー
アメリカ合衆国83713アイダホ州ボイジー　ウェスト・パーム・ストリート　11082